

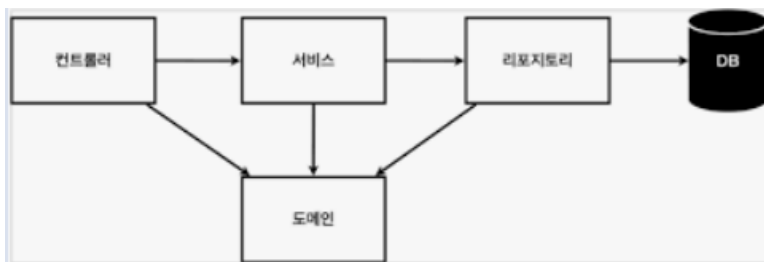
스프링 초기 설정 ver2.

프로젝트 - 큰 흐름



전체적인 구조

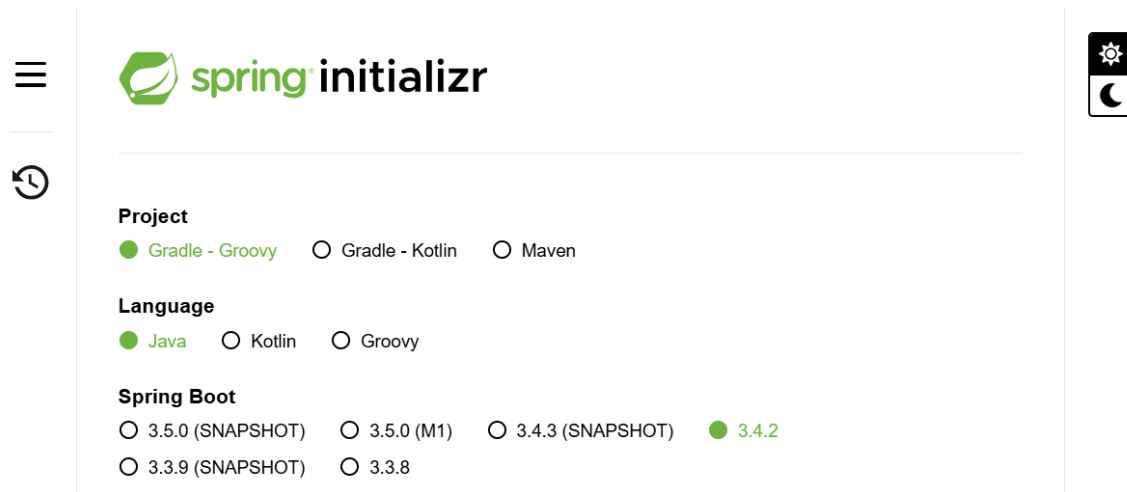
- **Controller** : 들어온 요청 / 응답을 보고 어떤 서비스를 호출할지를 결정
- **Service** : 비즈니스 로직 / 어떤 작업을 할지를 결정
- **Repository** : 서비스가 필요로 하는 데이터를 가져오는 역할 (DAO)



- **Domain** : 데이터 / 핵심 개념들을 나타내는 객체들 집합을 의미
- **DAO(Data Access Object)** : DB 데이터에 접근하기 위한 객체
- **DTO(Data Transfer Object)** : 데이터 교환을 위한 Object → 로직을 갖지 않는 객체로 getter, setter랑 필드 정도만 가진 클래스를 의미한다. 쉽게 말해서 데이터 뭉탱이로 담아서 넘기는 객체임

스프링 부트 설정

사이트 : <https://start.spring.io/>



The image shows the Spring Initializr web form. It has a sidebar on the left with a hamburger menu and a refresh icon. The main area has the 'spring initializr' logo and a settings panel on the right with a sun/moon icon. The settings panel includes sections for Project, Language, and Spring Boot, each with radio button options.

Project

☒ Gradle - Groovy ☐ Gradle - Kotlin ☐ Maven

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 3.5.0 (SNAPSHOT) ☐ 3.5.0 (M1) ☐ 3.4.3 (SNAPSHOT) ☒ 3.4.2

☐ 3.3.9 (SNAPSHOT) ☐ 3.3.8

- **Gradle-Groovy**

- **Language : 자바**

- **Spring Boot : 3.4.2**

→ (SNAPSHOT), (M1) 등은 아직 기능 개발중이거나 업데이트되어 수정될 가능성이 있는 파일들이라 들었음

Project Metadata

Group	com.beyond3
Artifact	yyGang
Name	yyGang
Description	beyond13 Team3 pj2
Package name	com.beyond3.yyGang
Packaging	<input checked="" type="radio"/> Jar <input type="radio"/> War
Java	<input checked="" type="radio"/> 23 <input type="radio"/> 21 <input type="radio"/> 17

- Group : com.beyond3
- artifact : yyGang
- packaging : Jar
- Java23

Dependencies

ADD ...

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Web Services WEB

Facilitates contract-first SOAP development. Allows for the creation of flexible web services using one of the many ways to manipulate XML payloads.

MariaDB Driver SQL

MariaDB JDBC and R2DBC driver.

Lombok DEVELOPER TOOLS

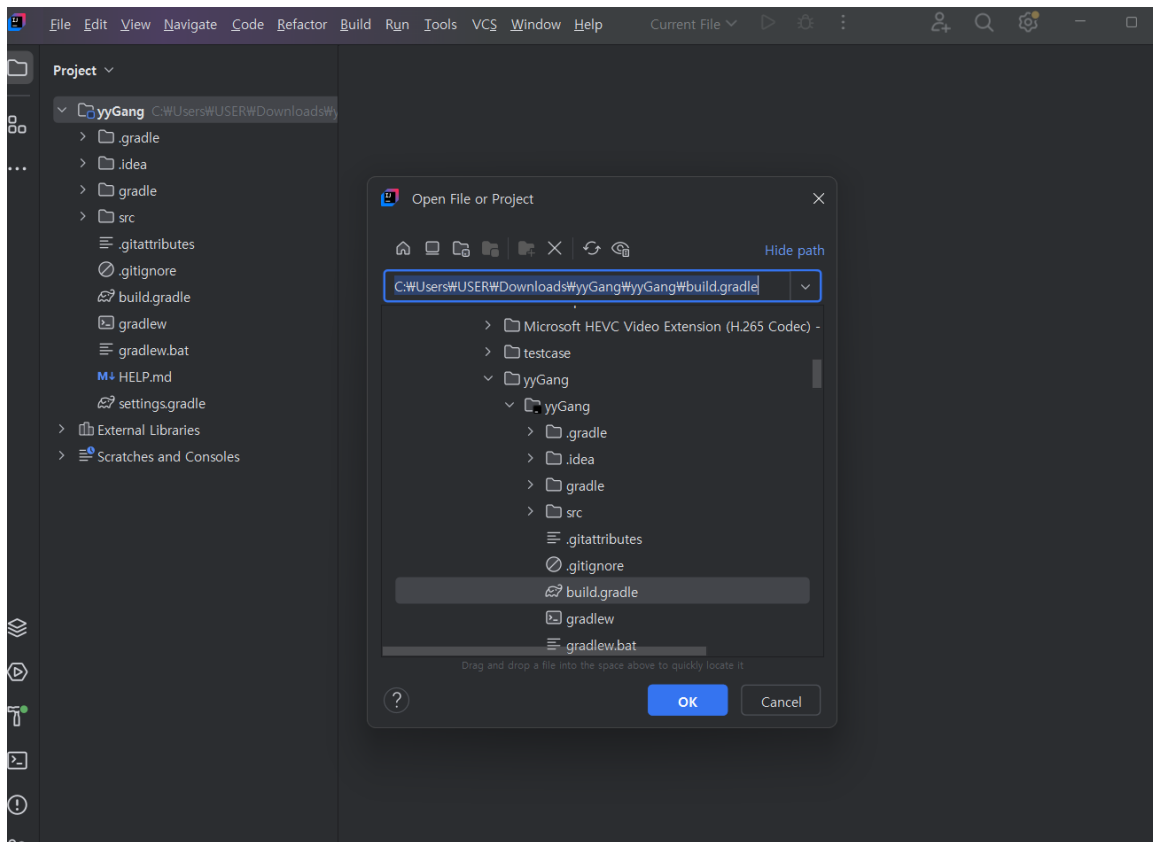
Java annotation library which helps to reduce boilerplate code.

- 우측 상단 ADD... 클릭
- Spring web , Spring web Service , MariaDB Driver , Lombok 차례로 검색해서 추가

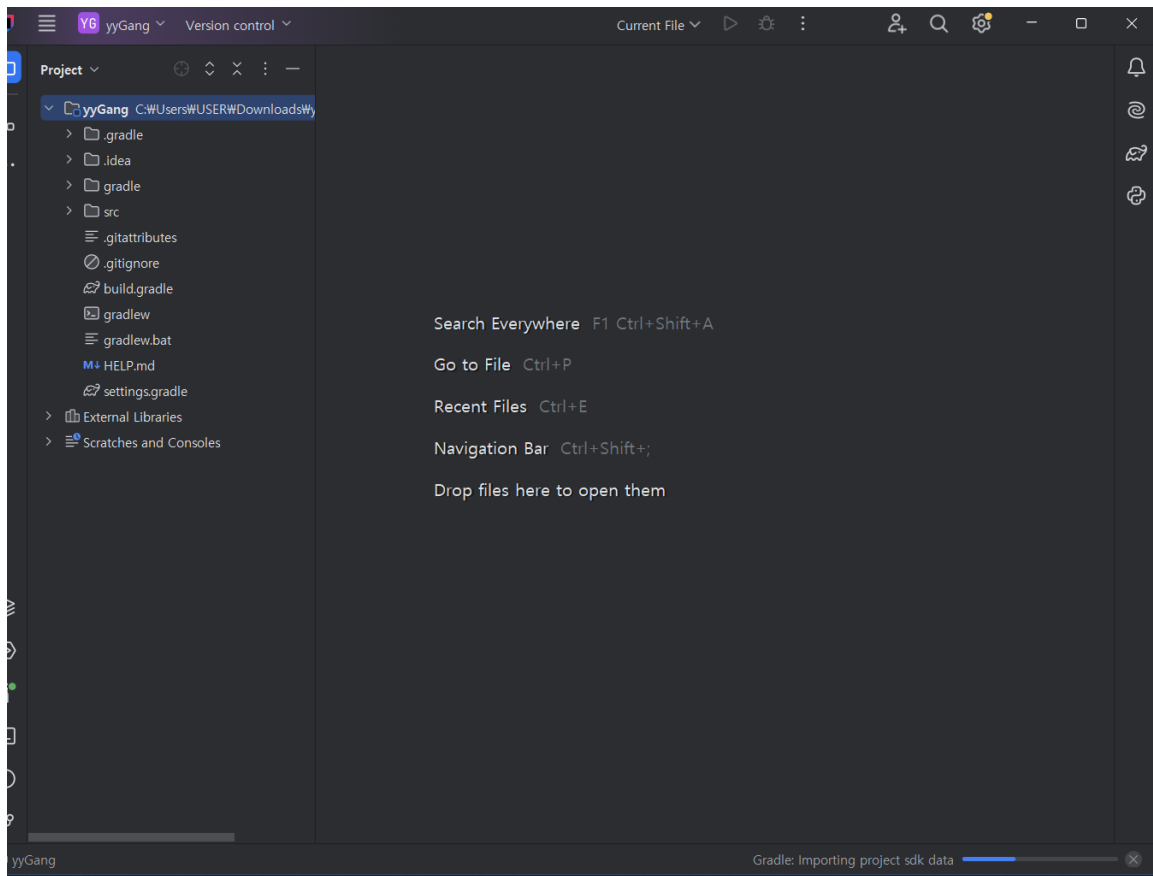
GENERATE 해주기

→ 압축 풀기 → 원하는 위치에 저장

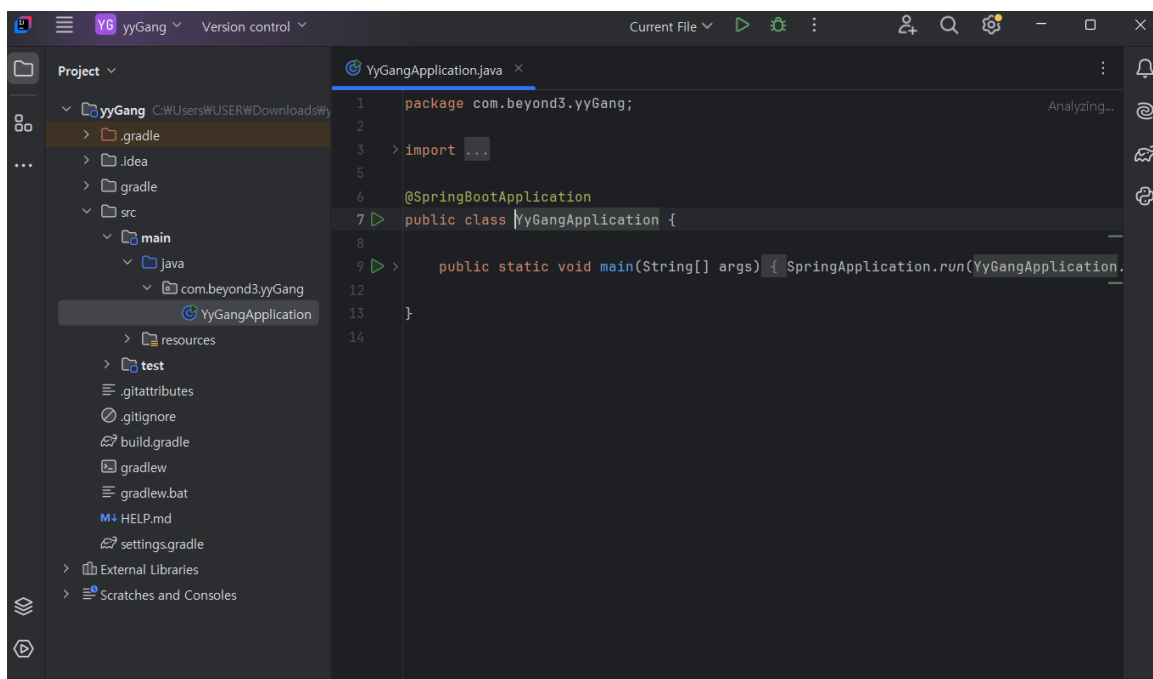
→ 인텔리 제이 → File → Open → 파일 저장 위치(yyGang) → build.gradle → OK → 프로젝트로 열기



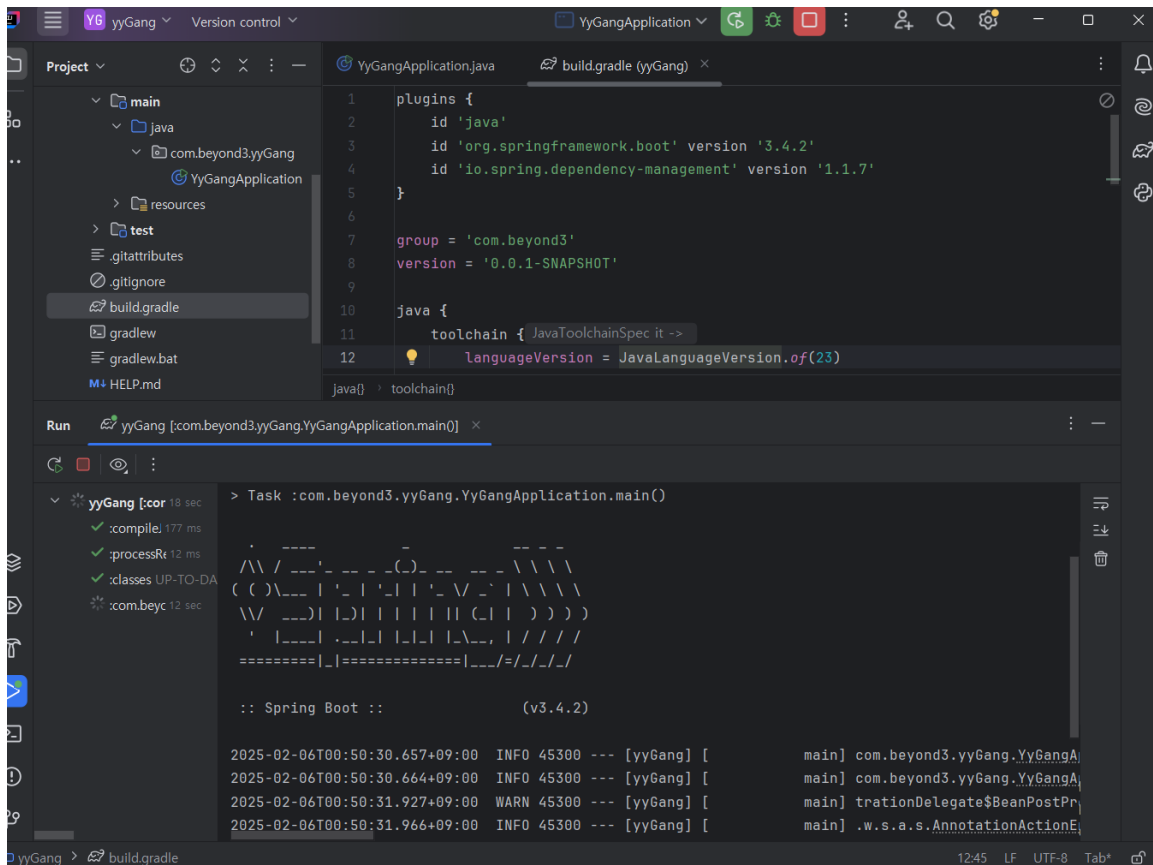
→ 생성하고 라이브러리 다운 받는데 시간이 좀 걸리는 편(우측 하단 파란색 로딩 표시 다 끝날 때까지 기다리기)



→ 다 됐으면 src → main → java → com.beyond3.yyGang.YyGangApplication 클릭 → 실행



→ 이런거 뜨면 성공



→ 크롬 열어서 <http://localhost:8080> 입력



Whitelabel Error Page

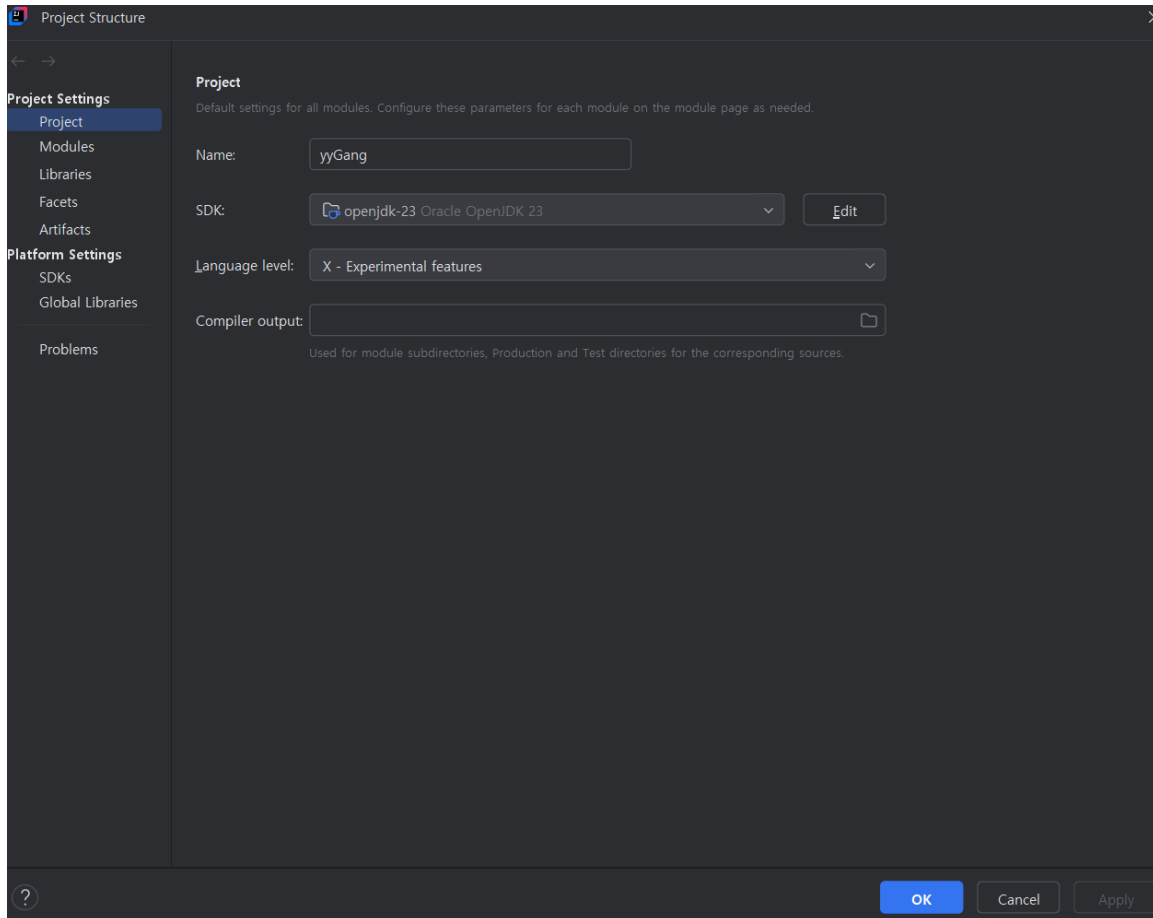
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Feb 06 01:03:50 KST 2025

There was an unexpected error (type=Not Found, status=404).

→ 이 화면 뜨면 성공

연결 안될 경우 - 자바 버전이 맞지 않는 경우



→ File → project Structure → SDK 버전 확인 → 위에서 java 23 버전 쓰려고 했으니까 SDK도 open jdk 23이어야 호환됨

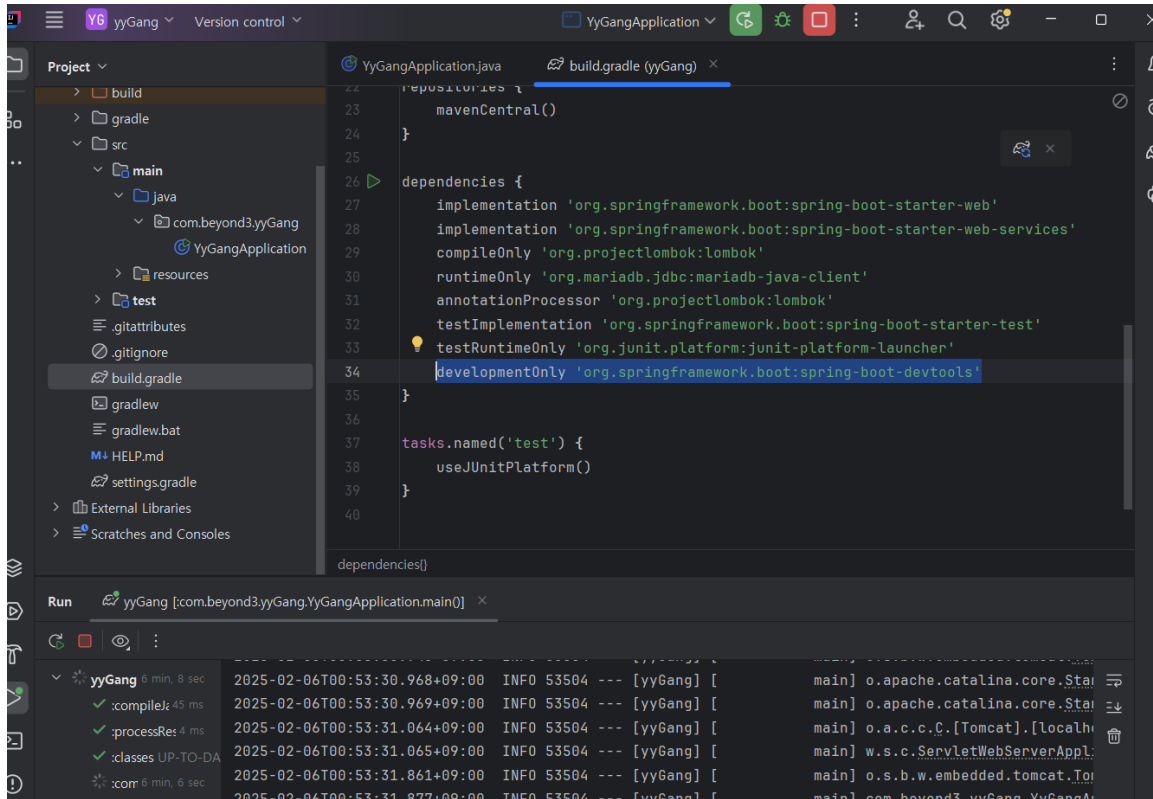
그 외 연결 안되는 경우 말해주면 해결하러 가겠습니다.

추가 설정하면 좋대요

바로바로 수정 사항 웹 페이지에 적용하는 법

- 스프링 개발 시 수정하나 하고 서버를 다시 띄우고 하는 번거로움을 거쳐야함
- Dependency에 devtools를 추가하면 새로고침 시 수정사항이 바로바로 적용됨

```
dependencies {
    developmentOnly 'org.springframework.boot:spring-boot-dev
```

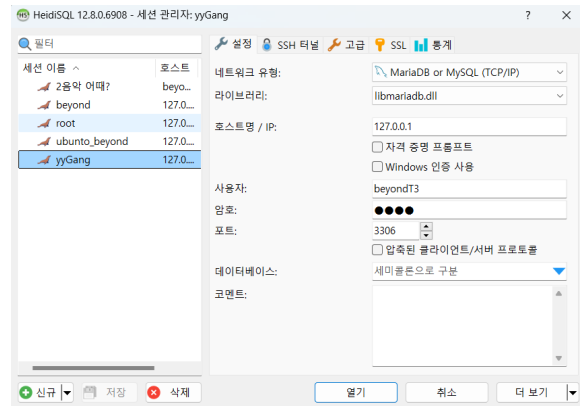
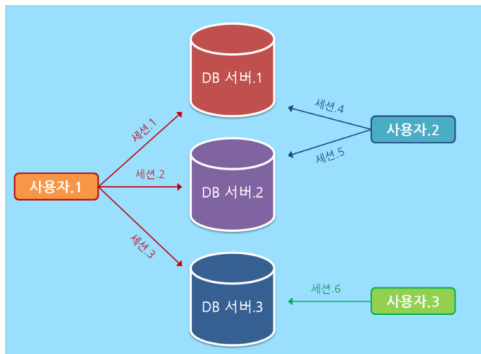


→ 이후 File -Settings → Advanced Settings → Compiler → Allow auto-make to start even if developed application is currently running 선택

→ Build, execution, deployment → Compiler → Build Project Automatically 선택

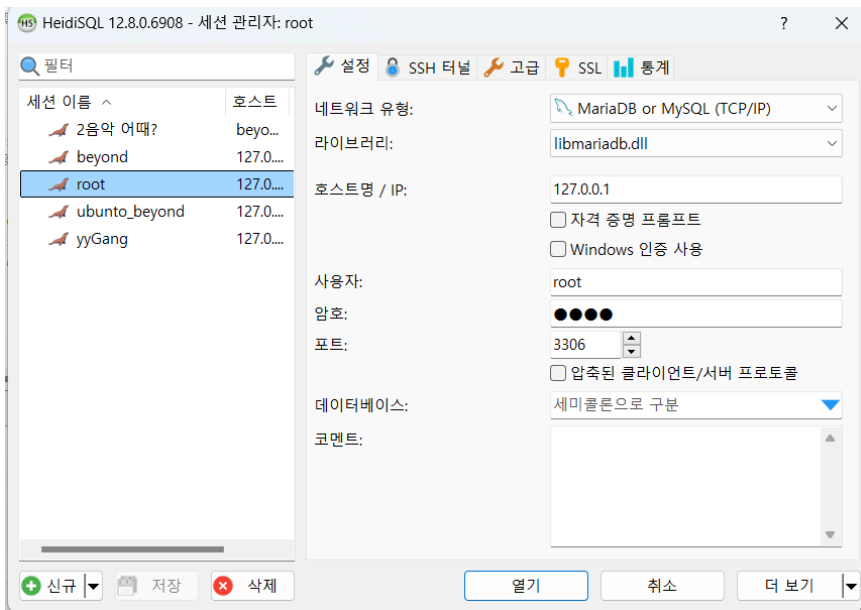
마리아 DB 연동

데이터 베이스 기초

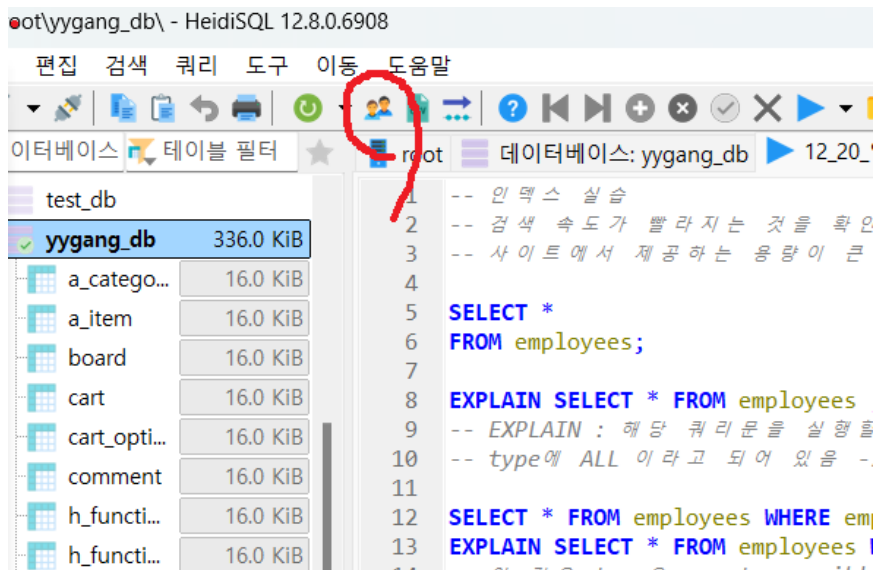


- **세션** : 쉽게 말해 사용자와 DB의 연결을 의미함.
- **사용자** : DB에 접속하려는 개인 or 애플리케이션
- **호스트** : 접속하려고 하는 데이터베이스의 서버
- **포트** : 동일한 IP 내에서 서버가 둘 이상 있을 때 이들을 구분하기 위한 번호

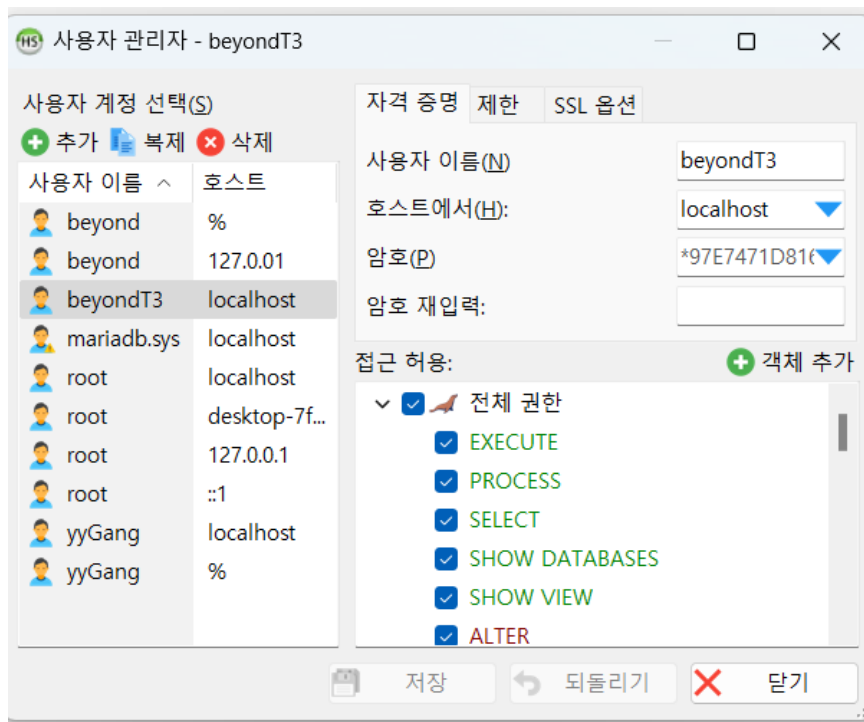
인텔리제이 - MariaDB 연결



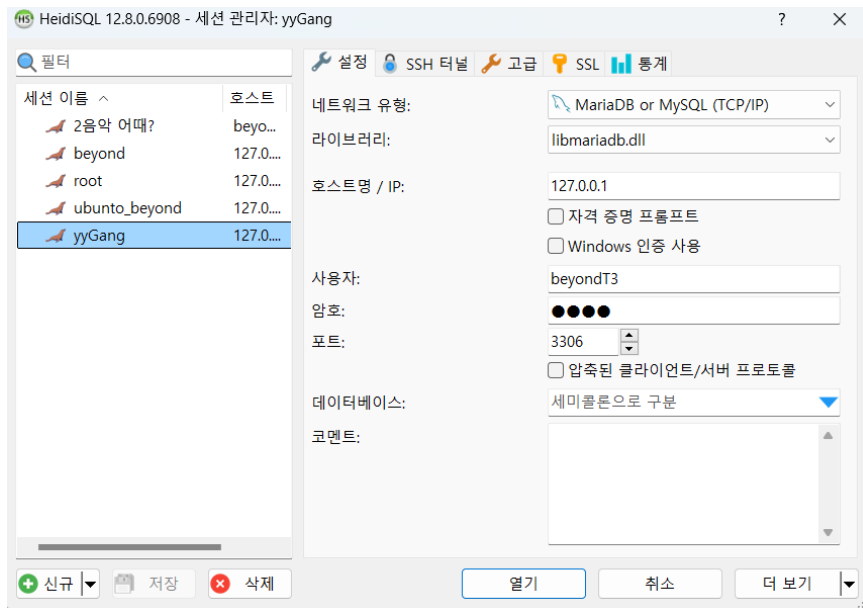
- root 사용자로 접속할 수 있는 세션에 접속한다. 필자는 root 세션을 따로 만들어 뒀고, 여기에 root 사용자로 자동으로 연결되게 만들어 두었기 때문에 root 세션에 root 사용자로 접속했다.



- 좌측 상단에 사람 표시를 누르면 사용자를 추가할 수 있다. root 사용자의 경우 “사용자 추가 권한”을 가진 사람이기 때문에 root 사용자로 접속한 것. → 사용자를 추가해보자!



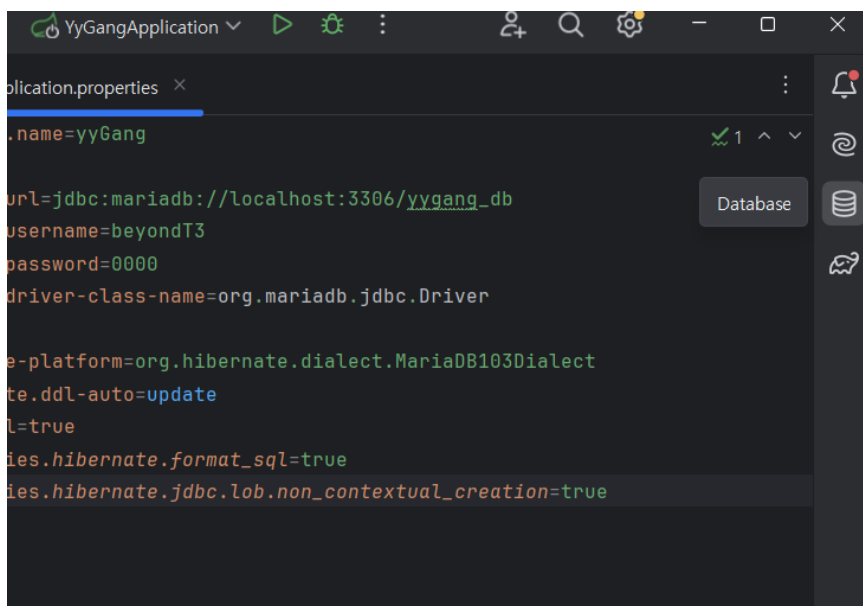
- 필자는 왼쪽 “추가”를 누르고 beyondT3로 사용자를 생성해주고, 호스트에서는 localhost로 지정해주었다. 또한 전체 권한을 부여해주었다. 암호는 본인 편한걸로 하자.
 - 호스트에서(H) : 사용자가 데이터베이스에 접속할 수 있는 위치를 의미함. 필자가 지정한 localhost 는 필자가 만들고 있는 beyondT3 사용자는 localhost위치(내 컴퓨터)에서만 데이터베이스에 접근할 수 있게 하겠다는 의미이다.



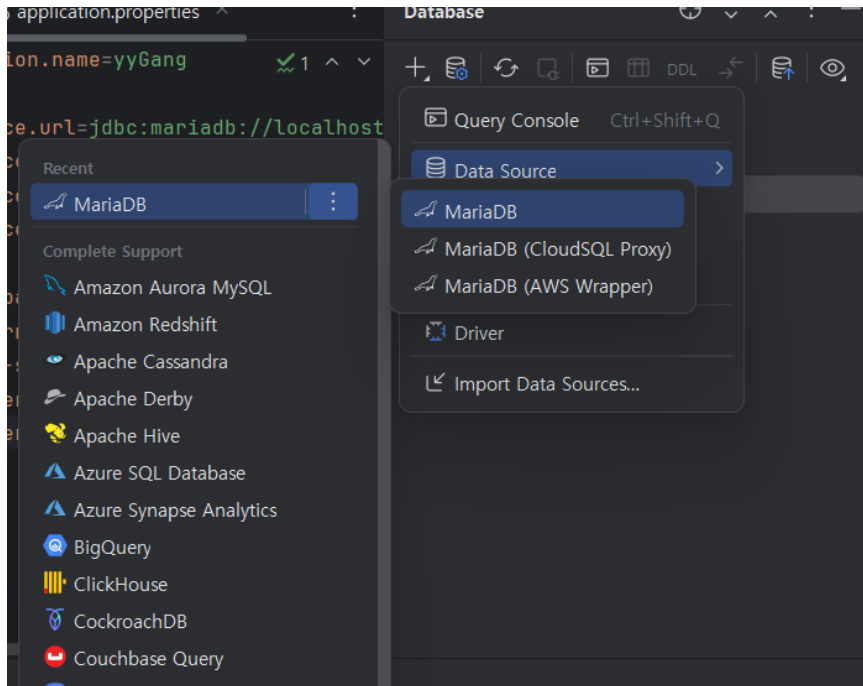
- 필자는 이후 신규 세션 yyGang을 만들고 접속 사용자를 방금 root 사용자에서 만들어 준 beyondT3 사용자로 지정해 세션을 저장한다. → 굳이 필요한 작업은 아니지만 이후 데이터 비교를 위해 혹시 몰라 추가했다.

그럼 본격적으로 인텔리제이 접속을 준비해보자!

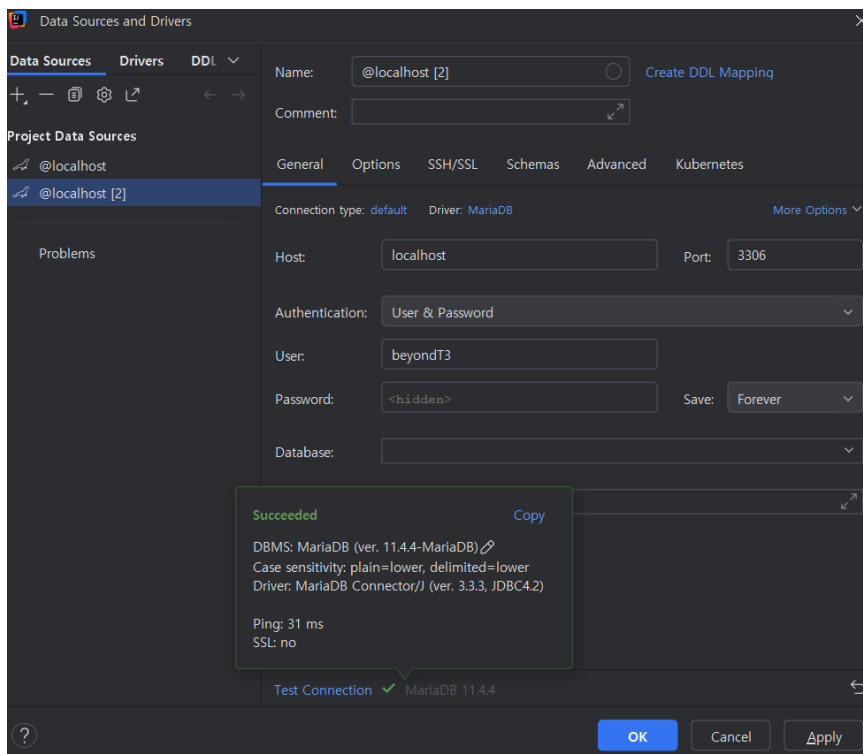
IntelliJ - MariaDB 연결



- 왼쪽 Database 클릭 or `ctrl + shift + 'a'` → database 선택

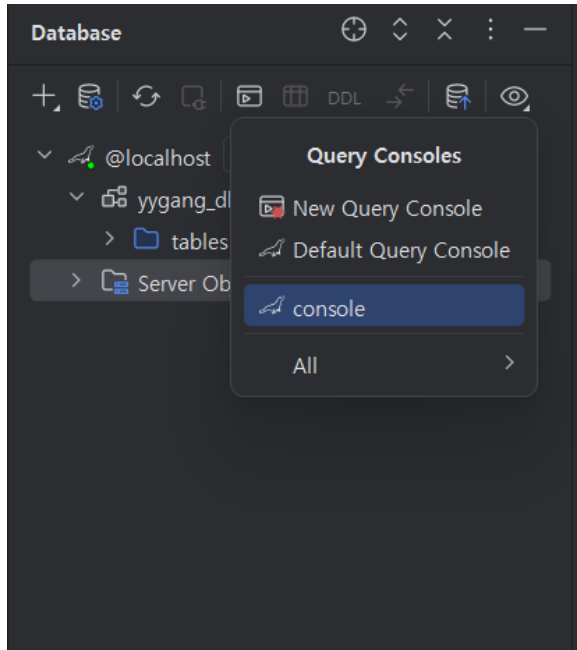


- "+" 누르기 → Data source → MariaDB 선택

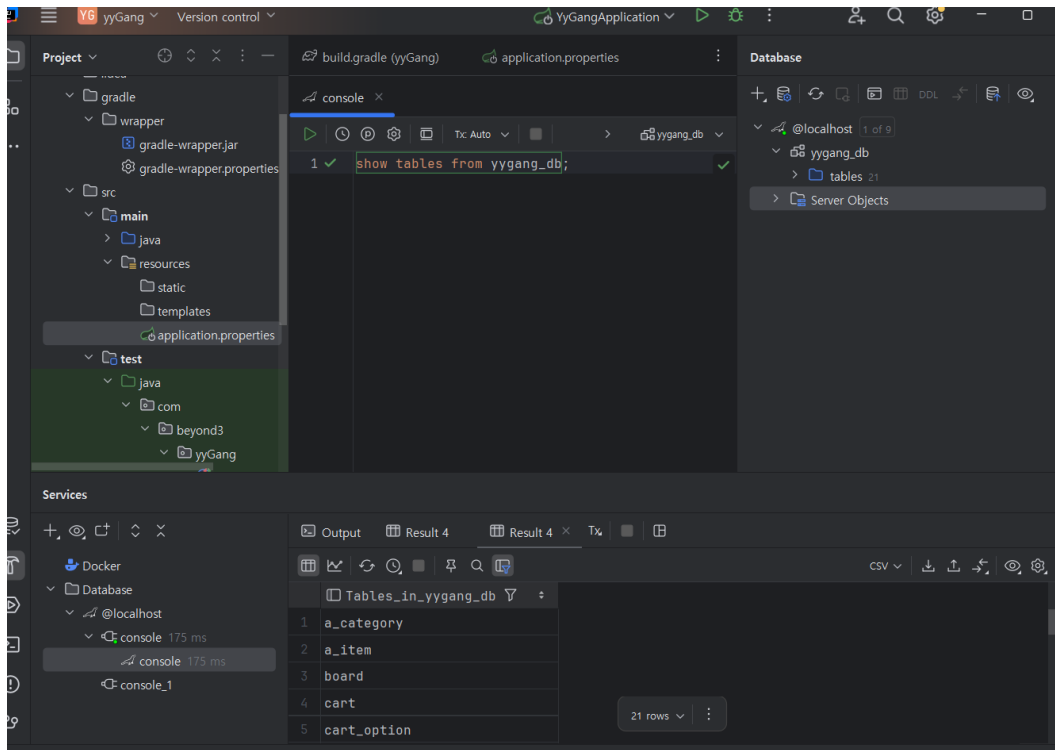


- User → 위에서 권한 다 부여한 beyondT3 사용자를 입력 + Password 입력

- 아래 Test Connection을 눌러서 **Succeeded** 가 나오면 성공! → **Apply** → **OK** 로 연결하기
- 어 왜 근데 세션은 따로 안만들어요? → 인텔리제이랑 마리아디비가 서로 연결될 때는 인텔리제이에서 자동으로 세션을 생성해서 연결해줌. 우리는 우리가 설정해준 권한을 가진 사용자를 통해 접속하기만 하면 된다!
- 필자는 Database에 영양갱 데이터베이스만 선택해서 넣었다.



- 마리아 디비랑 잘 연동됐는지 확인하자. console을 열면 heidiSQL에서 했던 방식 그대로 sql 쿼리문을 작성해서 바로바로 데이터 확인이 가능하다.
- 필자는 `show tables from yygang_db;` → 이걸로 데이터 베이스 내에 테이블들이 어떻게 있는지 테스트했다.



- 테스트 성공!!

JPA 설정해주기

build.gradle → 의존성 추가하기

- 프로젝트 빌드를 관리해주는 build.gradle → 프로젝트에서 사용할 라이브러리, 의존성, 프레임워크 등을 미리 정해주는 곳임. → 여기에다가 JPA 사용할거고 MariaDB 사용할것인지 설정해주어야겠지?

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.4.2'
    id 'io.spring.dependency-management' version '1.1.7'
}

group = 'com.beyond3'
version = '0.0.1-SNAPSHOT'

java {
```

```

    toolchain {
        languageVersion = JavaLanguageVersion.of(23)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'
    implementation 'org.springframework.boot:spring-boot-starter'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
}

tasks.named('test') {
    useJUnitPlatform()
}

```

- 위 코드에서

- `implementation 'org.springframework.boot:spring-boot-starter-data-jpa'`
- `runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'`

→ 이 두 구문이 있는지 확인하자. 각각은 스프링 부트에서 JPA를 사용하기 위한 스타터, MariaDB 데이터 베이스에 연결하기 위한 JDBC 드라이버이다. 필자는 이미 해당 구문이 build.gradle에 들어가 있었는데, 이는 위에서 했던 mariaDB 연동 과정에서 자동으로 들어간 듯 하다.

application.properties → 설정해주기

- 애플리케이션 설정 관리 파일

```
spring.application.name=yyGang

spring.datasource.url=jdbc:mariadb://localhost:3306/yygang_db
spring.datasource.username=beyondT3
spring.datasource.password=0000
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MariaDB103
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

- 구문을 전부 추가해주면 된다.
- `spring.datasource.url=jdbc:mariadb://localhost:3306/yygang_db`
 - 연결할 데이터베이스 호스트 이름 : 포트명/데이터베이스 이름 설정
- `spring.datasource.username=beyondT3`
`spring.datasource.password=0000`
 - maria DB에 접속할 사용자의 이름 및 암호를 입력

아래 구문들은 자동으로 테이블 업데이트 할지 등의 기타 설정들이다. 궁금하다면 따로 서칭해보자.

추가로 해야할 일

- 인텔리제이 HTTPs 이용해보기 → 이걸 추가 플러그인 다운 없이 바로바로 HTTP 날릴 수 있다고 함. 개발하면서 필요한 경우에만 날려보는 방식을 이용하자
- JPA가 제대로 연결되었는지 확인할 수 있는 테스트 케이스 작성 → 되냐 안되냐